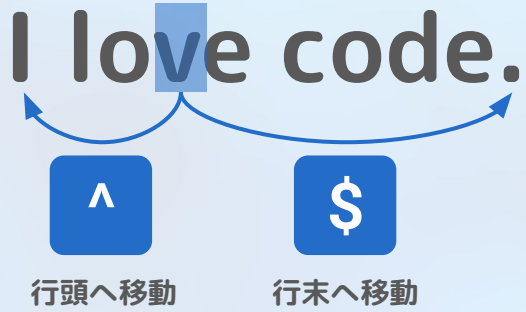
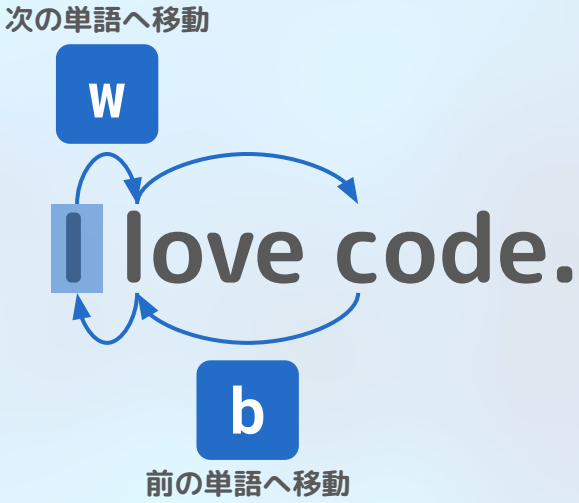
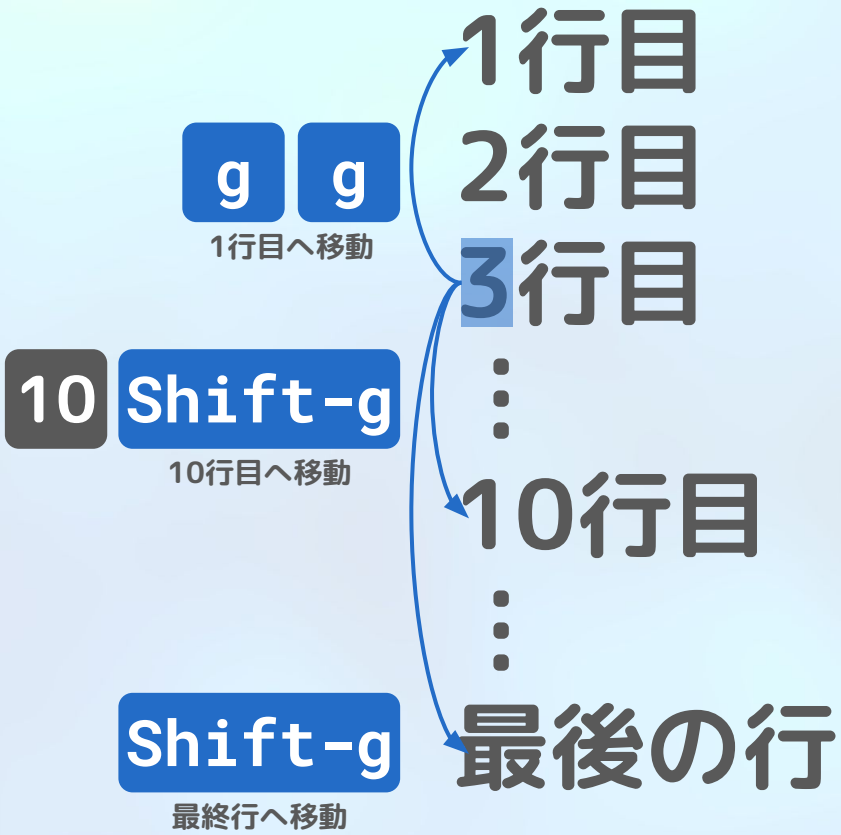
効率的なカーソル移動


I love code.
^
$
行頭へ移動
行末へ移動


次の単語へ移動
w
I love code.
b
前の単語へ移動

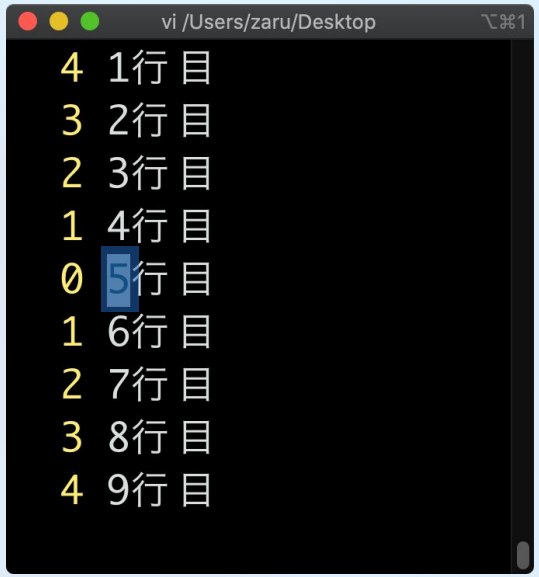

1行目
2行目
3行目
10行目
最後の行
g g
1行目へ移動
10 Shift-g
10行目へ移動
Shift-g
最終行へ移動

小ネタ

n行移動する時は相対行表示

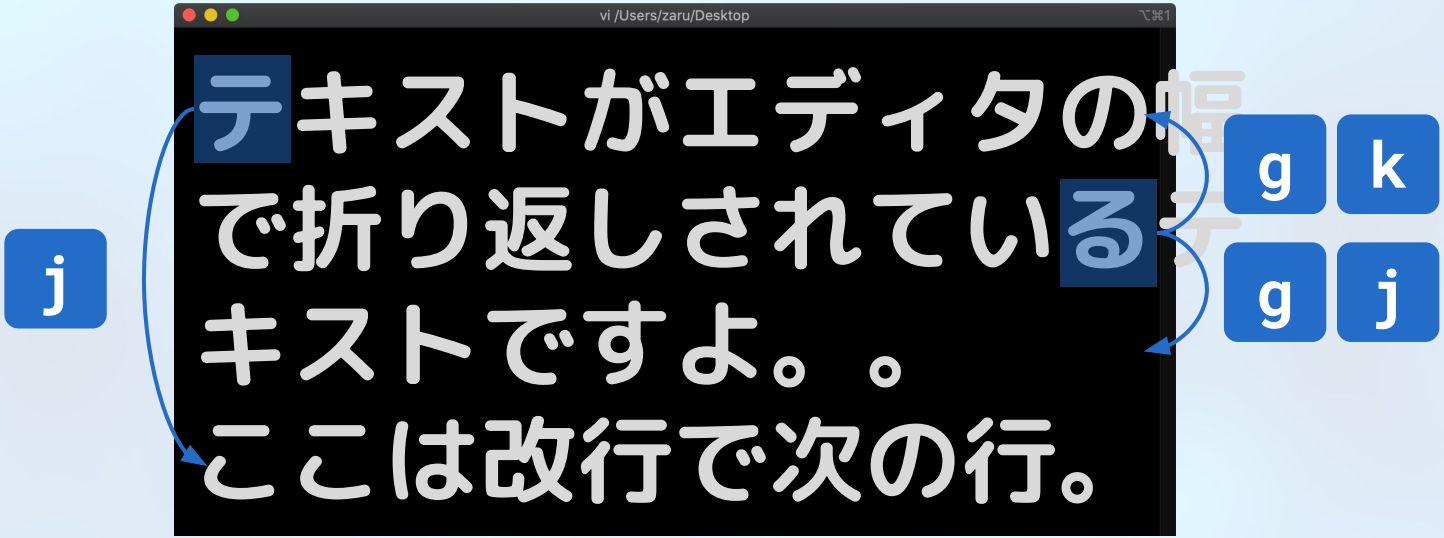
```
:set relativenumber
```

カーソル位置からの相対的な行数を表示してくれるので、Shift + n の指定が直感的



便利技

折り返しの中で上下移動



j や k は改行単位での行移動だが
g を前につけると見た目上の行移動が可能になる

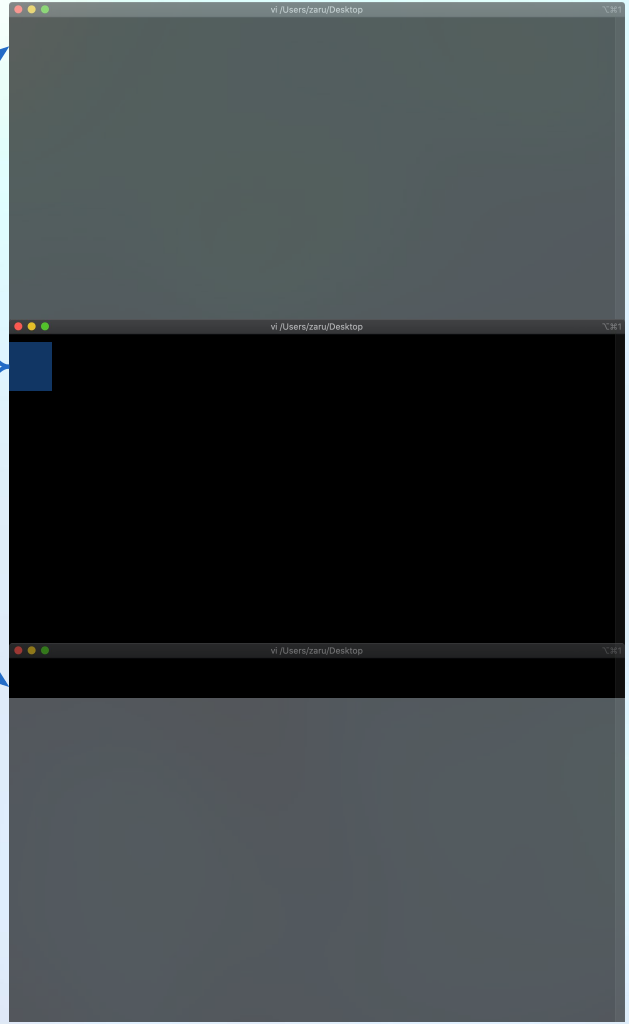
基本のページ移動

Ctrl-b

約1画面分
上に移動する

Ctrl-f

約1画面分
下に移動する



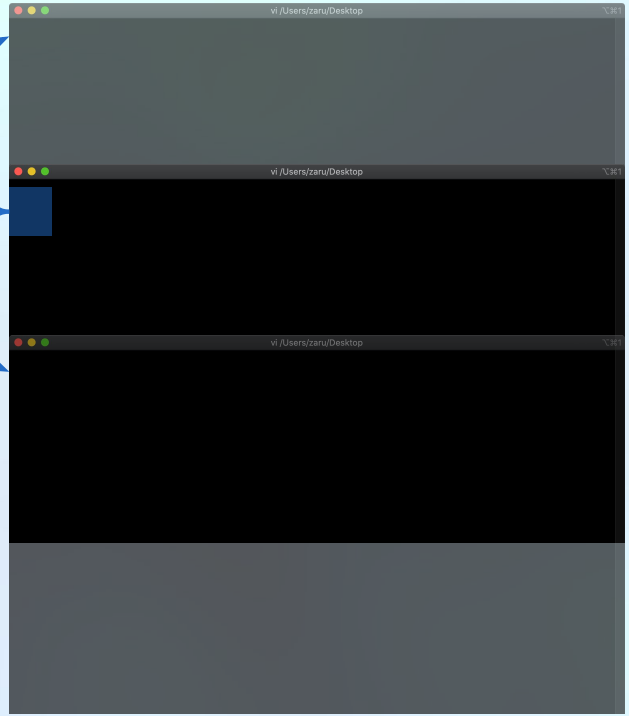
効率的なページ移動

Ctrl-u

約半画面分
上に移動する

Ctrl-d

約半画面分
下に移動する



画面上の
1番上にカーソル移動

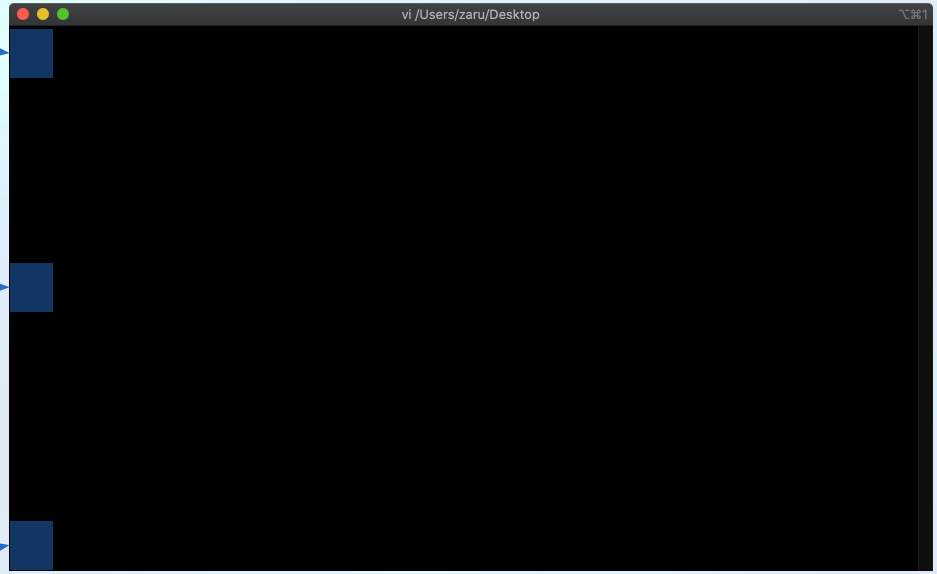
Shift-h

画面上の
真ん中にカーソル移動

Shift-m

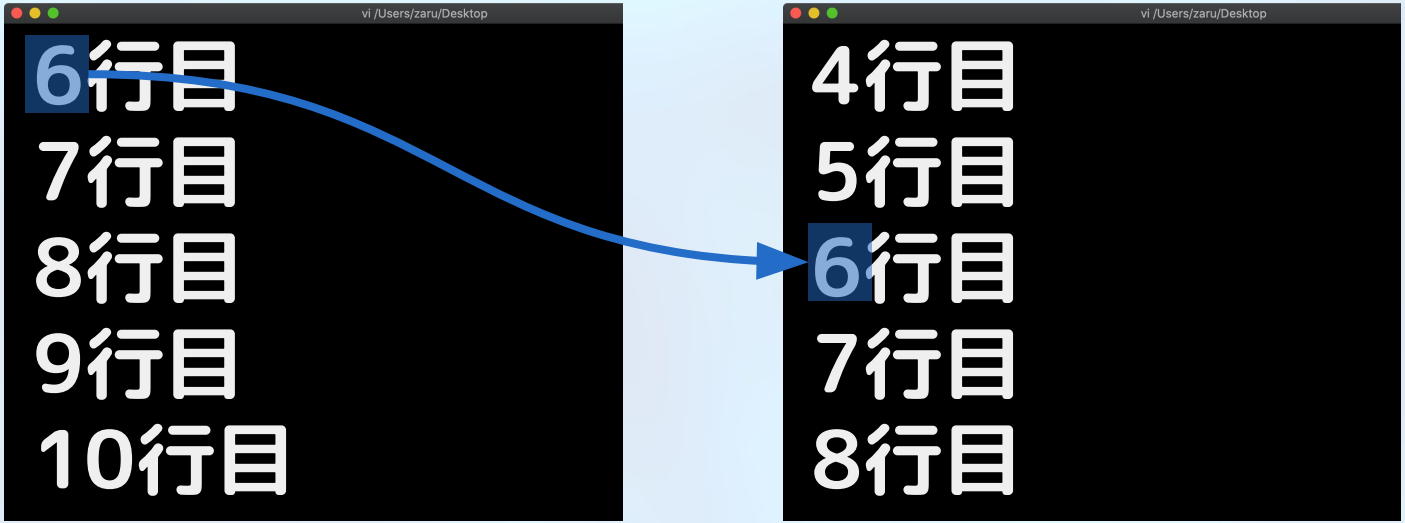
画面上の
1番下にカーソル移動

Shift-l



Z Z

カーソル位置を画面の
真ん中までスクロールする

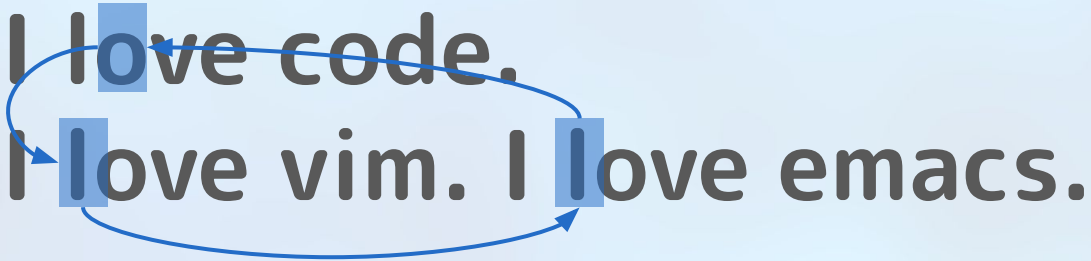


検索

/ スラッシュで検索コマンド

/love

/ の後に指定した文字を検索

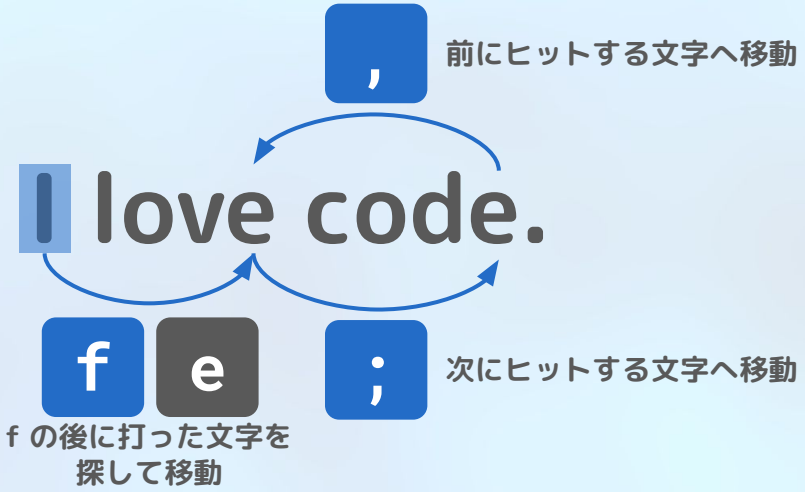


n

次の検索候補に移動

便利技

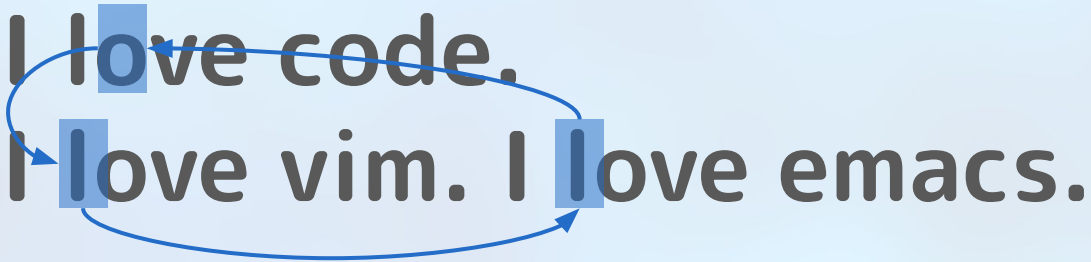
行の中で文字検索して移動



便利技

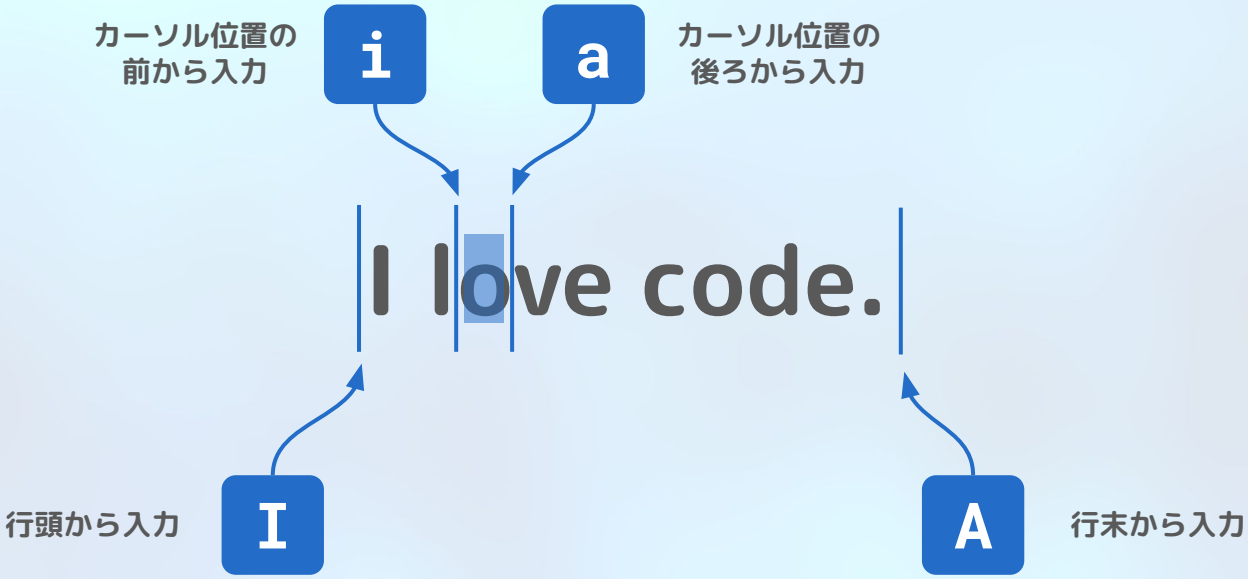
カーソル位置の単語を検索

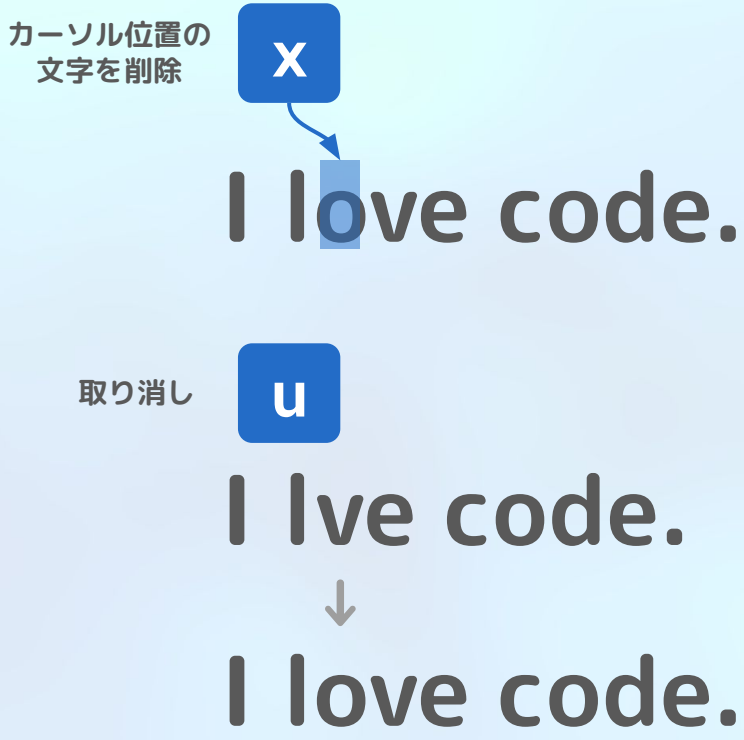
`*` カーソル位置の love を検索



`n` 次の検索候補に移動

基本の編集


カーソル位置の
前から入力
i
a
カーソル位置の
後ろから入力
I love code.
行頭から入力
I
A
行末から入力

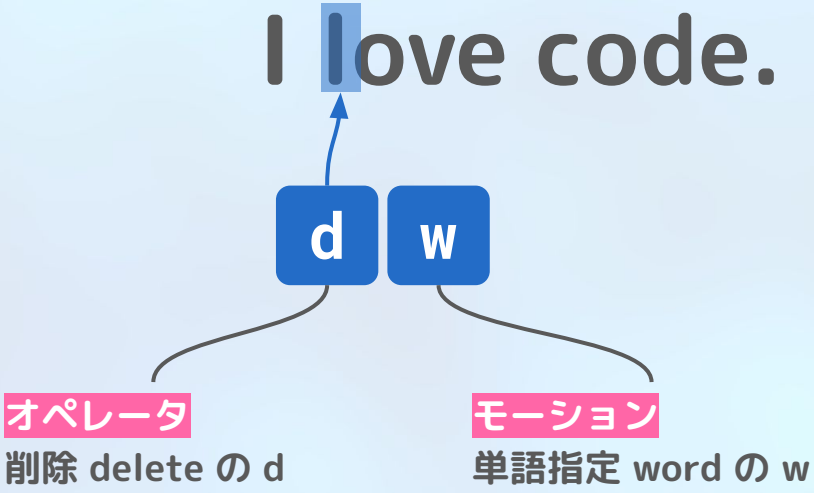

カーソル位置の
文字を削除
x
I love code.
取り消し
u
I lve code.
I love code.

もっと便利な編集
オペレータとモーション

例えば、単語を削除する

x で1文字ずつ削除するのに比べて非常に楽
これはオペレータとモーションを組み合わせている

オペレータはテキスト編集
モーションは範囲指定



オペレータ + モーション例

（テキストオブジェクト、あとで解説）

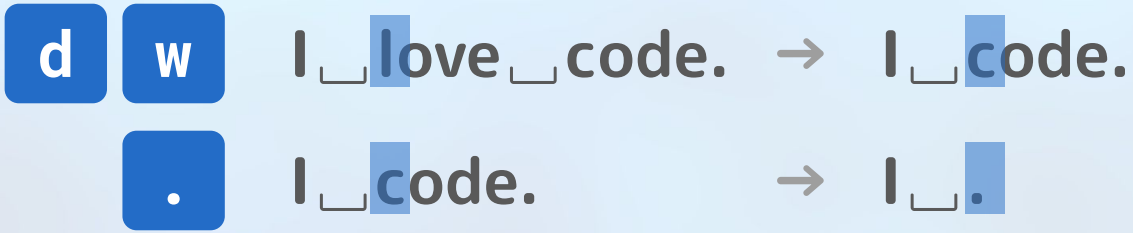
d	削除	+	aw	単語	=	カーソル位置の単語を削除
c	変更	+	iw	単語	=	カーソル位置の単語を削除して挿入モードへ
y	コピー	+	$	行末	=	行末までをコピー
>	インデント	+	j	次の行	=	今と次の行をインデント

ちなみにオペレータを2回繰り返すと、カーソル位置の行が対象範囲になる

・dd 現在の行を削除　・>> 現在の行をインデント

．で繰り返し

. は最後の変更を繰り返す



この場合は、最後の変更はカーソル位置の単語 (love) を削除した。これを繰り返すと、次のカーソル位置の単語は code なので、これが消える

テキストオブジェクト

文字のかたまりが
テキストオブジェクト

hoge 'piyo fuga'

v a' '' で囲まれた文字を選択する ('' を含む)

c i' '' で囲まれた文字を消して挿入モード

その他のテキストオブジェクト (a か i が付く)

aw	a"	a(	a{	a[	at
単語	""	()	{}	[]	タグ

小ネタ

i と a はノーマルモードとテキストオブジェクトでは意味が違う

ノーマルモードの i と、c オペレータが入力された後の i は違う。
モードによって同じキーでも役割が変わる。

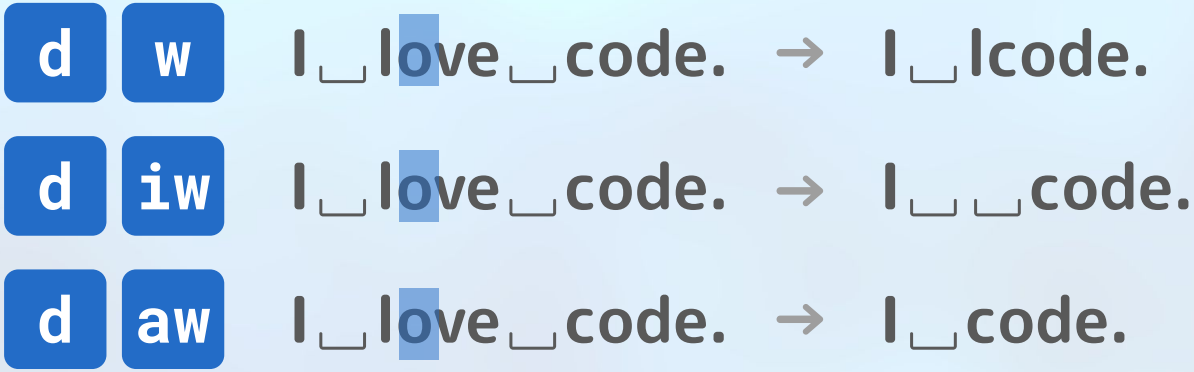
c オペレータ入力後にモーション入力を待っている状態は、iw とすることで
カーソル位置の単語を選択するテキスオブジェクトになる

ノーマルモードの i は insert、テキストオブジェクトの i は inner 的な感じ

小ネタ

テキストオブジェクトの i と a の違い

どれもカーソル位置の単語を削除するが、単語の選択範囲が違う

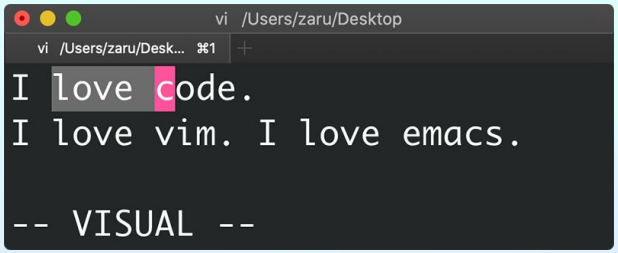


w はカーソル位置以降のみ（空白を含むかどうかはオペレータによって違う）
iw はカーソル前も含んだ単語で、空白を含まない
aw はカーソル前も含んだ単語で、空白を含む

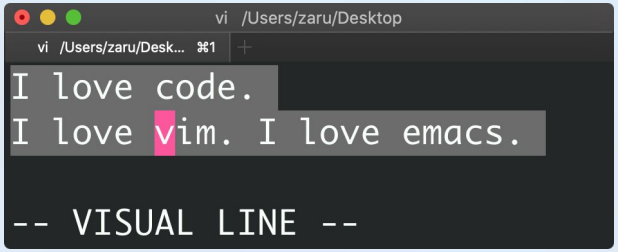
iw は c などの編集に向き、aw は d の削除に向いている

ビジュアルモード

ヴィジュアルモードは範囲選択できる



v ビジュアルモード（文字）



V ビジュアルモード（行）

Ctrl-v ビジュアルモード（矩形）

小ネタ

ESC と Ctrl-[は同じ

物理 ESC キーがない Mac ユーザ、ESC が遠くて押しにくい人は Ctrl-[が同じ効果を持っているので、こちらがオススメ。

ちなみに、Vim で Ctrl-[が ESC にエイリアスされているわけではなく、Ctrl キー自体が同時に入力したキーの ASCII が下位5ビット以外をゼロにする機能で、結果として [が ESC と同じ ASCII になるため。

ESC 0x1b で 2進数　11011

[は 0x5b で 2進数 1011011 → 0011011

Ctrl を押すと0になる